Apache Spark - Practice set 3

Total points 0/0

The respondent's email (**baburam@fusemachines.com**) was recorded on submission of this form.

Question 1:Which of the following transformation is not evaluated lazily ? *

- ○ filter()
- ○ repartition()
- [✓ None of the responses, all transformations are lazily evaluated.]
- ○ sample()
- ○ select()

Question 2:The code block down below intends to join df1 with df2 with inner join but it contains an error. Identify the error. d1.join(d2, d1.col(id) == df2.col(id), inner) *

- [✓ Quotes are missing. The correct query should be d1.join(d2, d1.col("id") == df2.col("id"), "inner")]
- ○ We cannot do inner join in spark 3.0, but it is in the roadmap.
- ○ The order is not correct. It should be like the following df1.join(d2, "inner", d1.col("id") === df2.col("id"))
- ○ There should be two === instead of ==. So the correct query is d1.join(d2, d1.col("id") === df2.col("id"), inner)

Question 3:If we want to create a constant string 1 as a new column ‘new_column’ in the dataframe df, which code block we should select ? *

- df.withColumn(“new_column”, 1)
- [✓ df.withColumn("new_column", lit("1"))]
- df.withColumnRenamed('new_column', lit(1))
- df.withColumn(new_column, lit(1))
- df.withColumn(“new_column”, lit(1))

Question 4:Choose the right code block in order to change add a new column to the following schema.schema = StructType([StructField("name",StringType(),True)]) *

- schema.append("new_column",StringType(),True)
- schema.add(StringType(), "new_column")
- schema.append(StringType(), "new_column")
- [✓ schema.add("new_column",StringType(),True)]

Question 5:What is the best description of a catalog ? *

- A JVM process in order to help garbage collection.
- [✓ It’s the interface for managing a metadata catalog of relational entites such as databases, tables, functions etc.]
- Logically equivalent of DataFrames.
- It is sparks core unit to parellize it’s workflow.

Question 6:Given the following statements regarding caching: *

```
1: The default storage level for a DataFrame is StorageLevel.MEMORY

2: The DataFrame class does have an unpersist() operation

3: The persist() method needs an action to load data from its source to materialize the DataFrame in cache
```

- [x] 2
- [ ] 1,3
- [ ] 1,2,3
- [ ] 1
- [ ] 1,2

Question 7:The code block shown below contains an error. Identify the error. *

```
1. def squared (s):
2.    return s*s
3.
4. spark.udf.register("square", squared) spark.range(1, 20).createOrReplaceTempView("test")
   spark.sql("select id, square(id) as id_squared from temp_test")
```

- [✓ We are not querying the right view. Correct code block should be: spark.sql("select id, square(id) as id_squared from test")]
- There is no error in the code.
- We are not refering to right database. Proper command should be: spark.sql("select id, squared(id) as id_squared from temp_test")
- We need to add quotes when using udf in sql. Proper usage should be: spark.sql("select id, "squared(id)" as id_squared from test")
- There is no column id created in the database.

Question 8:You have a need to sort a dataframe df which has some null values on column a. You want the null values to appear last, and then the rest of the rows should be ordered ascending based on the column a. Choose the right code block to achieve your goal. *

- ○ df.orderBy(asc("a"))
- ● df.orderBy(df.a.asc_nulls_last())
- ○ df.sort(asc_nulls_last(a))
- ○ df.orderBy(asc_nulls_last(a))
- ○ It is not possible to sort, when there are null values on the specified column.

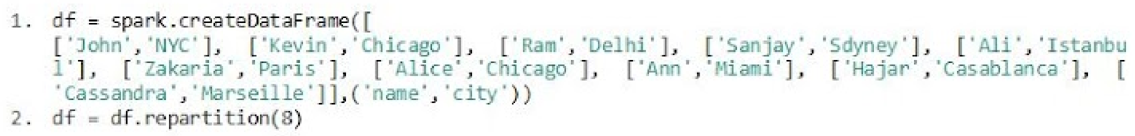
Question 9:Which of the followings are true for driver ? *

- ☐ Responsible for assigning work that will be completed in parallel.
- ☐ Is responsible for maintaining information about the Spark Application
- ☐ Controls physical machines and allocates resources to Spark Applications
- ☐ Executing code assigned to it
- ✓ Runs your main() function

Question 10:Which of the following code blocks merges two DataFrames df1 and df2 ? *

- df1.add(df2)
- df1.append(df2)
- df1.appendAll(df2)
- df1.addAll(df2)
- df1.merge(df2)
- [✓ df1.union(df2)]

Question 11:If we apply this code block df.rdd.getNumPartitions() What we will see ? Consider following dataframe *



```
df = spark.createDataFrame([
['John','NYC'],  ['Kevin','Chicago'],  ['Ram','Delhi'],  ['Sanjay','Sdyney'],  ['Ali','Istanbu
l'],  ['Zakaria','Paris'],  ['Alice','Chicago'],  ['Ann','Miami'],  ['Hajar','Casablanca'],  [
'Cassandra','Marseille']],('name','city'))
df = df.repartition(8)
```

- 4
- It is not a valid command, we will have an error
- 1
- [✓ 8]

Question 12:Select the code block which counts distinct number of “quantity” for each “invoiceNo” in the dataframe df. *

- [✓ df.groupBy("InvoiceNo").agg(expr("countDisctinct(Quantity)"))]
- df.groupBy("InvoiceNo").agg(expr(count(Quantity)))
- df.reduceBy("InvoiceNo").agg(expr("count(Quantity)"))
- df.groupBy(InvoiceNo).agg(expr(count(Quantity)))
- df.groupBy(InvoiceNo).agg(expr("count(Quantity)"))

Question 13:Which of the following code blocks changes the parquet file content given that there is already a file exist with the name that we want to write ? *

- df.write.option("compression", "snappy").save("path")
- df.format(“parquet”).mode("overwrite").option("compression", "snappy").save("path")
- [✓ df.write.format(“parquet”).option("compression", "snappy").path("path")]
- df.write.mode("overwrite").option("compression", "snappy").save("path")

Question 14:Choose valid execution modes in the following responses. *

- [✓ Standalone]
- [✓ Cluster]
- [✓ Client]
- [✓ Local]

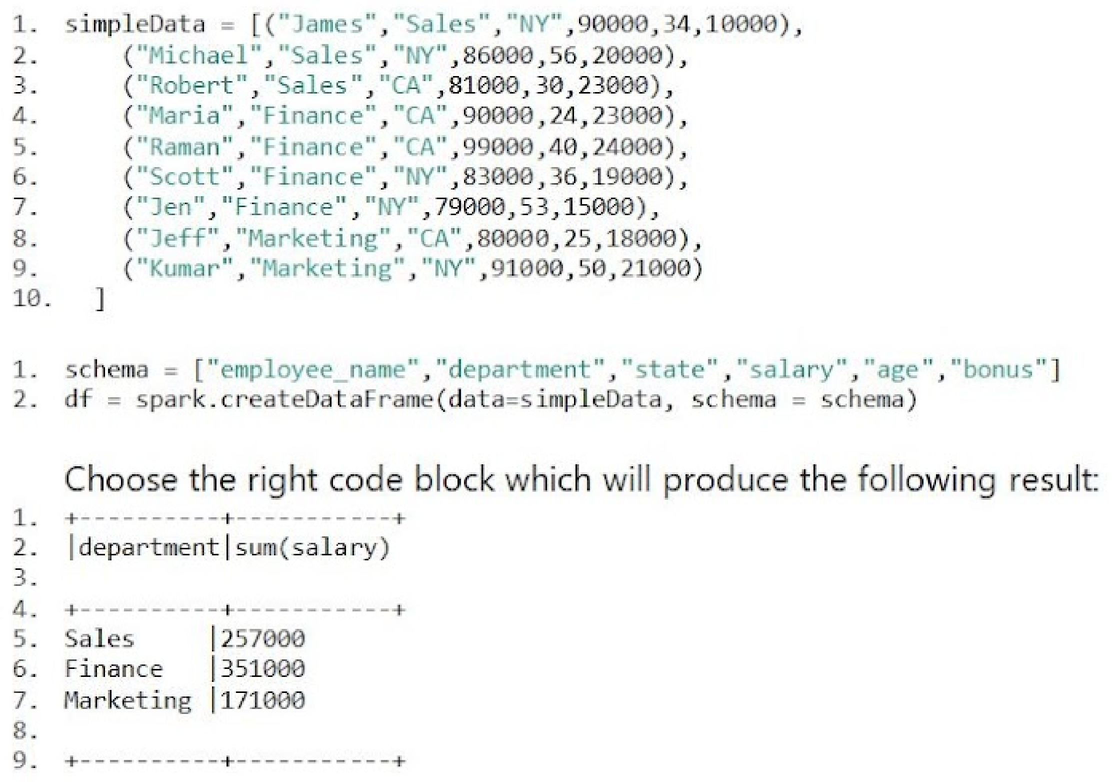
Question 15:At which stage Catalyst optimizer generates one or more physical plans ? *

- ○ Logical Optimization
- ○ Analysis
- [✓ Physical Planning]
- ○ Code Generation

Question 16:Which of the followings is NOT a useful use cases of spark ? *

- ○ Building, training, and evaluating machine learning models using MLlib
- ○ Performing ad hoc or interactive queries to explore and visualize data sets
- ○ Analyzing graph data sets and social networks
- [✓ Processing in parallel small data sets distributed across a cluster]

Question 17:Given an instance of SparkSession named spark, and the following DataFrame named df: *



```
simpleData = [("James","Sales","NY",90000,34,10000),
    ("Michael","Sales","NY",86000,56,20000),
    ("Robert","Sales","CA",81000,30,23000),
    ("Maria","Finance","CA",90000,24,23000),
    ("Raman","Finance","CA",99000,40,24000),
    ("Scott","Finance","NY",83000,36,19000),
    ("Jen","Finance","NY",79000,53,15000),
    ("Jeff","Marketing","CA",80000,25,18000),
    ("Kumar","Marketing","NY",91000,50,21000)
  ]
```

```
schema = ["employee_name","department","state","salary","age","bonus"]
df = spark.createDataFrame(data=simpleData, schema = schema)
```

Choose the right code block which will produce the following result:

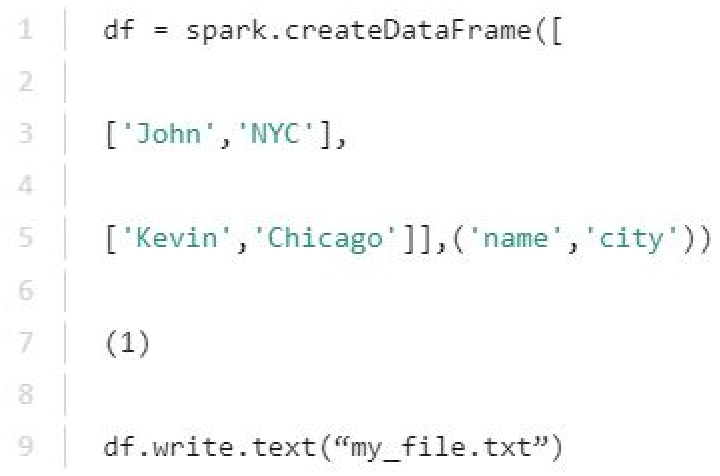
```
+----------+-----------+
|department|sum(salary)

+----------+-----------+
Sales     |257000
Finance   |351000
Marketing |171000

+----------+-----------+
```

- df.groupBy("department").sumAll("salary").show(truncate=False)
- df.groupBy("department").agg("salary").show(truncate=False)
- df.reduce("department").sum("salary").show(truncate=False)
- df.groupBy(department).sum(salary).show(truncate=False)
- [✓ df.groupBy("department").sum("salary").show(truncate=False)]

Question 18:There is a temp view named ‘my_view’. If I want to query this view within spark, which command I should choose ? *

- spark.read.view("my_view")
- spark.read.table("my_view")
- spark.read.table("global_temp.my_view")
- [✓ spark.read.view("my_view")]

Question 19:The code block shown below contains an error. The code block is intended to write a text file in the path. What should we add to part 1 in order to fix ? *



```
df = spark.createDataFrame([

['John','NYC'],

['Kevin','Chicago']],('name','city'))

(1)

df.write.text(“my_file.txt”)
```

- [✓ df.take()]
- df.drop("name")
- df.repartition(8)
- df.coalse(4)

Question 20:Which of the following code property is used for enabling adaptive query ? *

- spark.sql.optimize.adaptive
- spark.adaptive
- [✓ spark.sql.adaptive.enabled]
- spark.sql.adaptive
- spark.adaptive.sql

Question 21:Which of the following code blocks reads from a tsv file where values are separated with '\t' ? *

- [✓ spark.read.format("csv").option("header", "true").option("inferSchema", "true").option("sep", "\t").load(file)]
- spark.load.option("header", "true").option("inferSchema", "true").read(file)
- spark.read.option("header", "true").option("inferSchema", "true").option("sep", "true").toDf(file)
- spark.read.format("tsv").option("header", "true").option("inferSchema", "true").load(file)

Question 22:Which of the following operation is classified as a narrow transformation ? *

- ○ repartition()
- [✓ map()]
- ○ orderBy()
- ○ distinct()
- ○ collect()

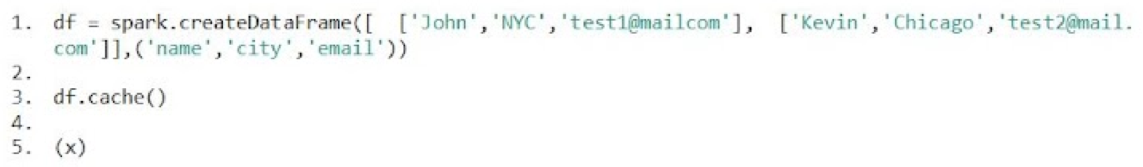
Question 23:What does the following property achieves in spark when enabled ? spark.sql.adaptive.skewJoin.enabled *

- ○ It allows you to dynamically select physical plans based on cost.
- ○ The goal of this property is to allow you to read only as much data as you need.
- [✓ Spark dynamically handles skew in sort-merge join by splitting (and replicating if needed) skewed partitions with this property enabled.]
- ○ It allows you to dynamically convert physical plans to RDDs

Question 24:The code block shown below intends to return a new DataFrame with column “old” renamed to “new” but it contains an error. Identify the error. df.withColumnRenamed(old, new) *

- You need to reverse parameters and add quotes. So correct code block is df.withColumnRenamed(“new”, “old”)
- [✓ You need to add quotes; correct usage is df.withColumnRenamed(“old”, “new”)]
- We need to add ‘col’ to specifiy that it’s a column. df.withColumnRenamed(col(“new”), col(“old”))
- WithColumnRenamed is not a valid fonction , we need to use df.withColumnRenamed(“new”, “old”)

Question 25:For the following dataframe if we want to fully cache the dataframe immediately, what code block should replace (x) ? *



```
df = spark.createDataFrame([  ['John','NYC','test1@mailcom'],  ['Kevin','Chicago','test2@mail.
com']],('name','city','email'))

df.cache()

(x)
```

- df.take(1)
- df.cache()
- df.takeAll()
- [✓ df.persist()]
- df.count()

Question 26:Which of the following statements about Spark accumulator variables is true? *

- ○ You can define your own custom accumulator class by extending org.apache.spark.util.AccumulatorV1 in Java or Scala or pyspark.AccumulatorParams in Python.
- [✓ The Spark UI displays named accumulators used by your application.]
- ○ Accumulators provide a immutable variable that a Spark cluster cannot update on a per-row basis.
- ○ For accumulator restarted tasks will update the value in case of a failure.
- ○ In transformations, each task’s update can be applied only once if tasks or job stages are re-executed.

Question 27:Let’s suppose that we have a dataframe df with a column ‘today’ which has a format ‘YYYY-MM-DD’. You want to add a new column to this dataframe ‘week_later’ and you want it’s value to be one week after to column ‘today’. Select the correct code block. *

- ○ df.withColumn("week_later", col("today") + 7))
- ○ df.withColumn("week_later", week_add(col("today"), 7))
- ○ df.withColumn(date_add(col("today"), 7), “week_later”)
- [✓ df.withColumn(week_later, date_add(col("today"), 7))]
- ○ df.withColumn("week_later", date_add(col("today"), 7))

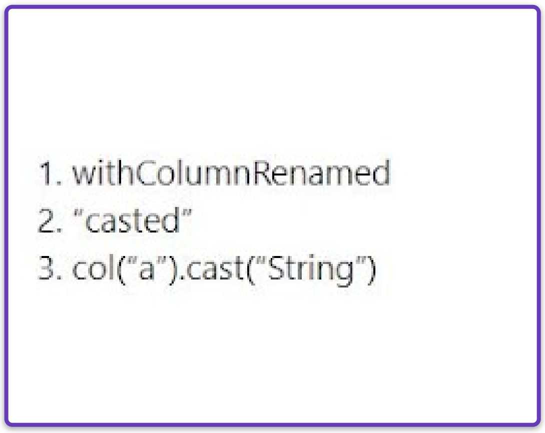
Question 28:If we want to store RDD as serialized Java objects in the JVM and if the RDD does not fit in memory, store the partitions that don't fit on disk, and read them from there when they're needed, which storage level we need to choose ? *

- ○ MEMORY_ONLY_2
- ○ MEMORY_AND_DISK
- [✓ MEMORY_AND_DISK_SER]
- ○ MEMORY_AND_DISK_2

Question 29:What will cause a full shuffle knowing that dataframe 'df' has 2 partitions ? *

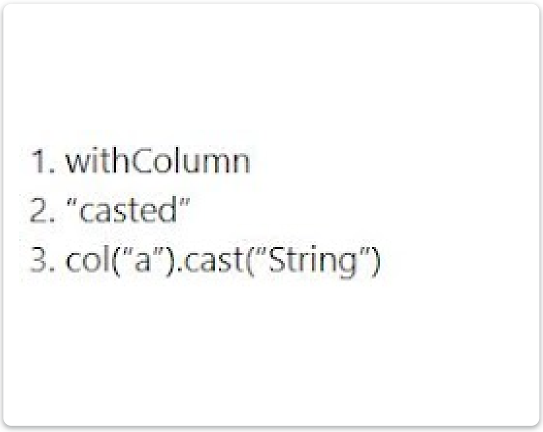
- ○ All of them will cause a full shuffle.
- ○ df.coalse(4)
- [✓ df.repartition(12)]

Question 30:The code block shown below should return a new DataFrame with a new column named “casted” who’s value is the string equivalent of column “a” which is a integer column, also this dataframe should contain all the previously existing columns from DataFrame df. Choose the response that correctly fills in the numbered blanks within the code block to complete this task. Code block: df._1_(_2_, _3_) *



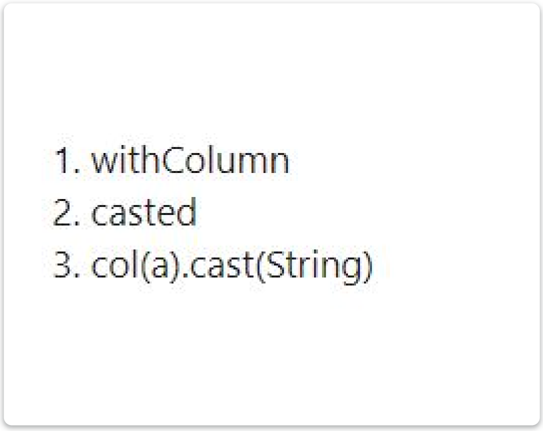
1. withColumnRenamed
2. “casted”
3. col(“a”).cast(“String”)

[✓ Option 1]



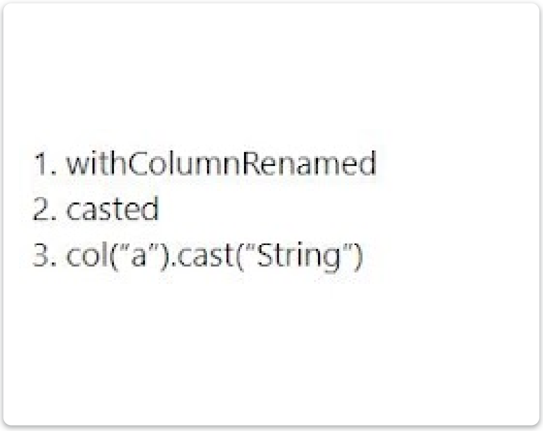
1. withColumn
2. “casted”
3. col(“a”).cast(“String”)

Option 2



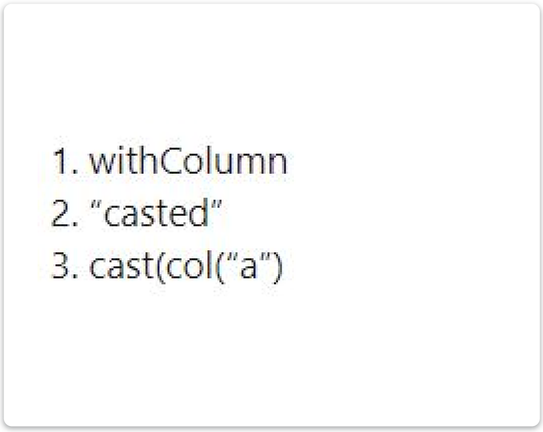
1. withColumn
2. casted
3. col(a).cast(String)

Option 3

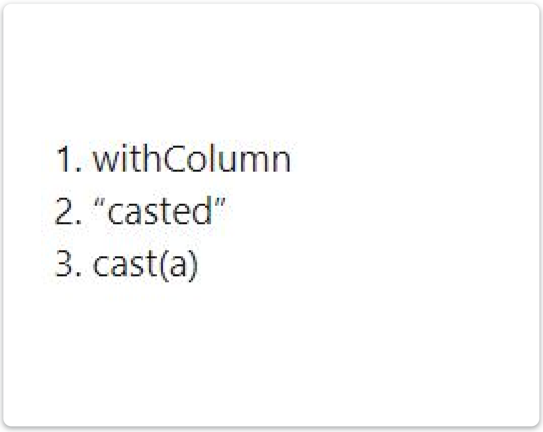


1. withColumnRenamed
2. casted
3. col(“a”).cast(“String”)

Option 4



1. withColumn
2. “casted”
3. cast(col(“a”)



1. withColumn
2. “casted”
3. cast(a)

- ○ Option 5
- ○ Option 6

Question 31:Choose the correct code block to unpersist a table named ‘table’. *

- ○ spark.uncacheTable(table)
- [✓ spark.catalog.uncacheTable(“table”)]
- ○ spark.catalog.uncacheTable(table)
- ○ spark.uncacheTable(“table”)

Question 32:Which of the following statement is NOT true for broadcast variables ? *

- ☐ The canonical use case is to pass around a small large table that does fit in memory on the executors.
- ☐ You can define your own custom broadcast class by extending org.apache.spark.util.BroadcastV2 in Java or Scala or pyspark.AccumulatorParams in Python.
- [✓ Broadcast variables are shared, immutable variables that are cached on every machine in the cluster instead of serialized with every single task.]
- [✓ It provides a mutable variable that a Spark cluster can safely update on a per-row basis.]
- ☐ It is a way of updating a value inside of a variety of transformations and propagating that value to the driver node in an efficient and fault-tolerant way.

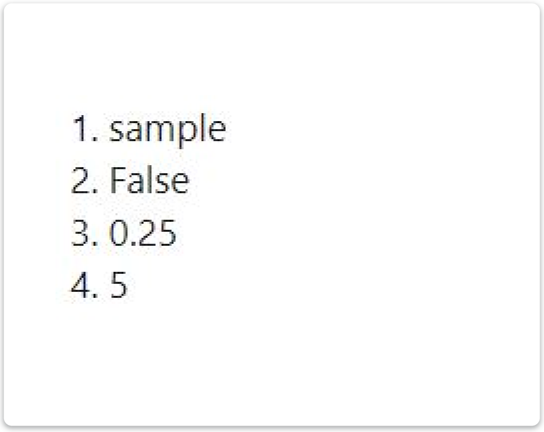
Question 33:What is the first thing to try if garbage collection is a problem ? *

- ○ Descrease java heap space size
- [✓ First thing to try if garbage collection is a problem is to use serialized caching]
- ○ Persist objects in deserialized form

Question 34:Which of the following describes a stage best ? *

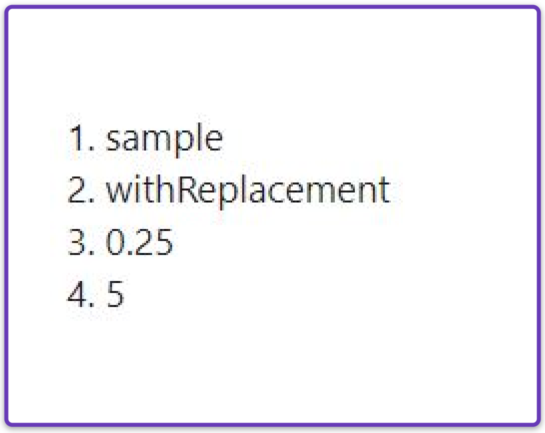
- ○ A unit of work that will be sent to one executor.
- ○ An external service for acquiring resources on the cluster (e.g. standalone manager, Mesos, YARN)
- [✓ A physical unit of execution which is a sequence of tasks that can all be run together in parallel without a shuffle.]
- ○ User program built on Spark. Consists of a driver program and executors on the cluster.
- ○ A process launched for an application on a worker node, that runs tasks and keeps data in memory or disk storage across them.

Question 35:The code block shown below should return a new DataFrame with 25 percent of random records from dataframe df with replacement. Choose the response that correctly fills in the numbered blanks within the code block to complete this task. Code block: df._1_(_2_, _3_, _4_) *



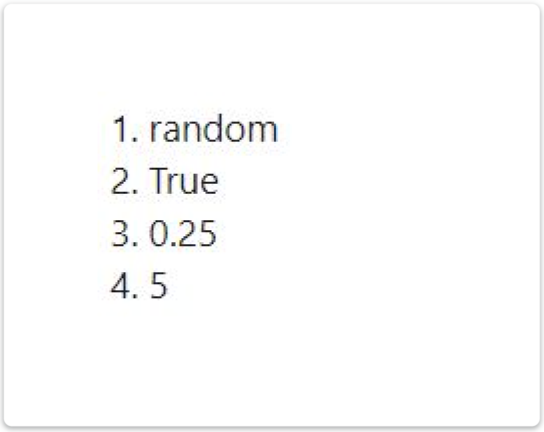
1. sample
2. False
3. 0.25
4. 5

Option 1



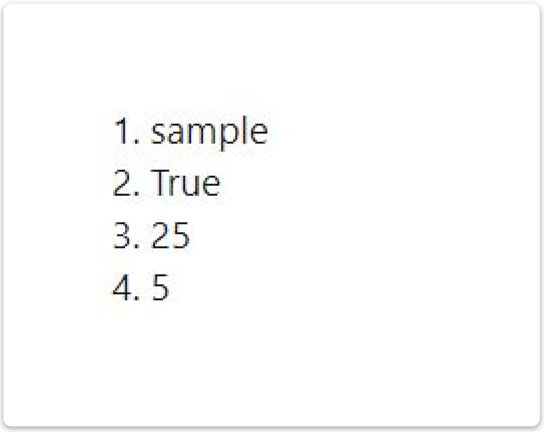
1. sample
2. withReplacement
3. 0.25
4. 5

[✓ Option 2]



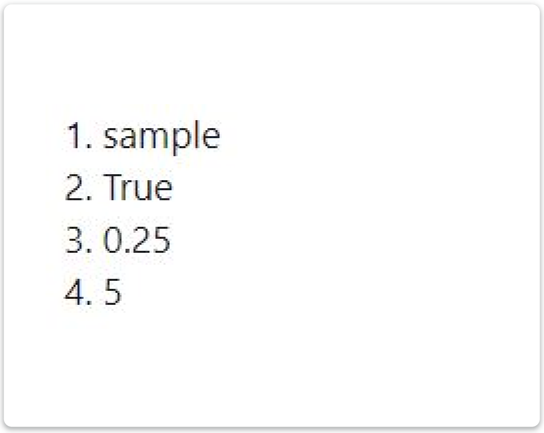
1. random
2. True
3. 0.25
4. 5

Option 3

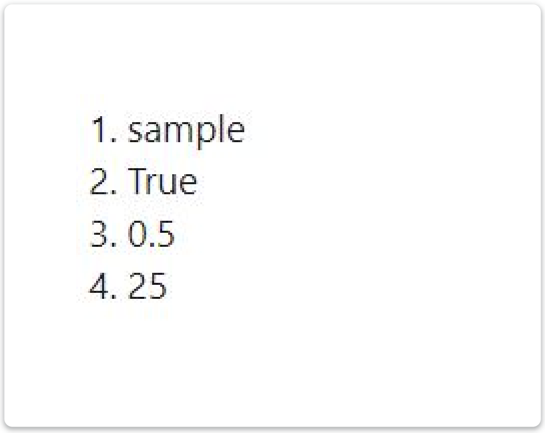


1. sample
2. True
3. 25
4. 5

Option 4


1. sample
2. True
3. 0.25
4. 5

Option 5


1. sample
2. True
3. 0.5
4. 25

Option 6

Question 36:The following statement will create a managed tabledataframe.write.saveAsTable("unmanaged_my_table") *

TRUE

[✓ FALSE]

Question 37:If spark is running in cluster mode, which of the following statements about nodes is correct ? *

There is one single worker node that contains the Spark driver and all the executors.

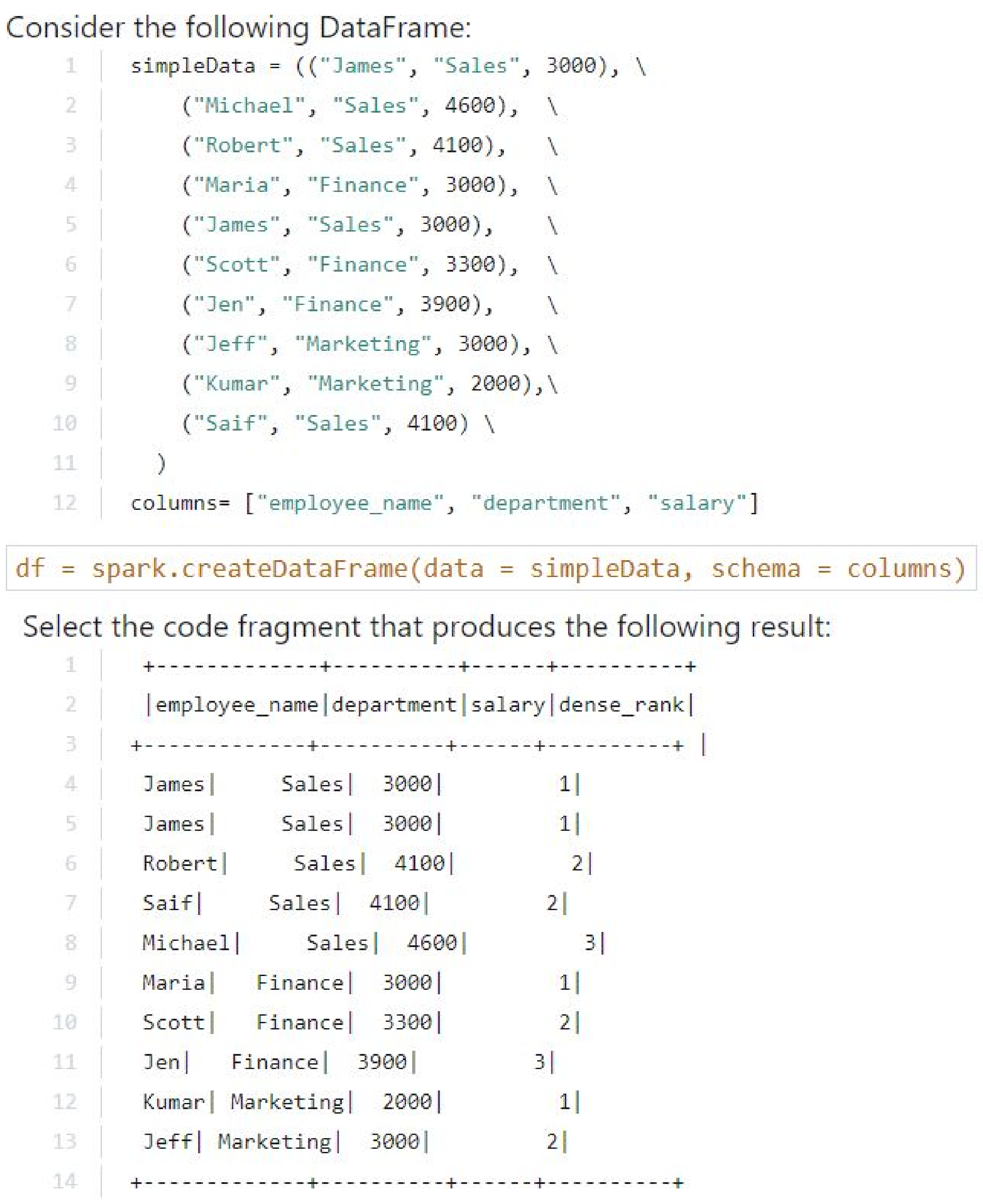
The spark driver runs in worker node inside the cluster.

[✓ Each executor is running JVM inside of a cluster manager node.]

There is always more than one node.

There are less executors than total number of nodes

Question 38: *



Consider the following DataFrame:

```
simpleData = (("James", "Sales", 3000), \
    ("Michael", "Sales", 4600),  \
    ("Robert", "Sales", 4100),   \
    ("Maria", "Finance", 3000),  \
    ("James", "Sales", 3000),    \
    ("Scott", "Finance", 3300),  \
    ("Jen", "Finance", 3900),    \
    ("Jeff", "Marketing", 3000), \
    ("Kumar", "Marketing", 2000),\
    ("Saif", "Sales", 4100) \
  )
columns= ["employee_name", "department", "salary"]
```

```
df = spark.createDataFrame(data = simpleData, schema = columns)
```

Select the code fragment that produces the following result:

```
 +-------------+----------+------+----------+
 |employee_name|department|salary|dense_rank|
+-------------+----------+------+----------+ |
 James|     Sales|  3000|         1|
 James|     Sales|  3000|         1|
 Robert|     Sales|  4100|         2|
 Saif|     Sales|  4100|         2|
 Michael|     Sales|  4600|         3|
 Maria|   Finance|  3000|         1|
 Scott|   Finance|  3300|         2|
 Jen|   Finance|  3900|         3|
 Kumar| Marketing|  2000|         1|
 Jeff| Marketing|  3000|         2|
+-------------+----------+------+----------+
```

- [✓ windowSpec =Window.partitionBy("department").orderBy("name")
  df.withColumn("rank",rank().over(windowSpec)) .show()]
- windowSpec=Window.partitionBy("department").orderBy("salary")
  df.withColumn("rank",rank().over(“windowSpec”)) .show()
- windowSpec = Window.partitionBy("department").orderBy("salary")
  df.withColumn("rank",rank().over(windowSpec)).show()

○ from pyspark.sql.functions import dense_rank windowSpec = Window.partitionBy("department").orderBy("salary") df.withColumn("dense_rank",dense_rank().over(“windowSpec”)) .show()

○ from pyspark.sql.functions import dense_rank windowSpec = Window.partitionBy("department").orderBy("salary") df.withColumn("dense_rank",dense_rank().over(windowSpec)) .show()

Question 39:Which property is used to to allocates jobs to different resource pools to achieve resources scheduling within an application ? *

○ There is no need to set a property since spark is by default capable of resizing

○ Dynamic allocation

[✓ Fair Scheduler]

Question 40:When the property spark.sql.optimizer.dynamicPartitionPruning.enabled is set to true, what optimization happens in spark ? *

[✓ It allows you to dynamically switch join strategies.]

○ You to read only as much data as you need.

○ It allows you to dynamically select physical plans based on cost.

○ Spark dynamically handles skew in sort-merge join by splitting (and replicating if needed) skewed partitions with this property enabled.

Question 41:Choose the correct code block to broadcast dfA and join it with dfB ? *

[✓ dfA.join(broadcast(dfB), dfA.id == dfB.id)]

dfB.join(broadcast(“dfA”), dfA.id == dfB.id)

dfA.join(broadcast(“dfB”), dfA.id == dfB.id)

dfB.join(broadcast(dfA), dfA.id == dfB.id)

Question 42:Given the following dataframe df = spark.createDataFrame([("A", 20), ("B", 30), ("D", 80)],["Letter", "Number"])We want to store the sum of all “number”'s in a variable ‘result’. Choose the correct code block in order to achieve this goal. *

result = df.groupBy().sum()

result = df.reduce().sum().collect()[0][0]

result = df.groupBy().sum().collect()

[✓ result = df.groupBy().sum().collect()[0][0]]

Question 43:If spark is running in client mode, which of the following statement about is NOT correct ? *

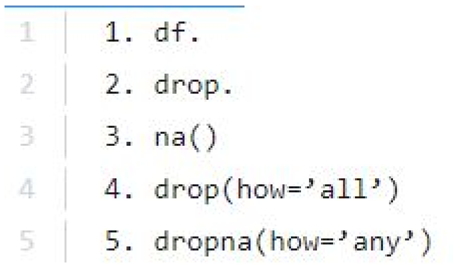
The entire spark application is run on a single machine.

In this mode worker nodes reside in the cluster.

Machines that who runs the driver called gateway machines or edge nodes.

[✓ Spark driver remains on the client machine that submitted the application.]

Question 44:We want to drop any rows that how a null value. Choose the correct order in order to achieve this goal. *



```
1. df.
2. drop.
3. na()
4. drop(how='all')
5. dropna(how='any')
```

- ○ 1, 4
- ○ 1, 2, 3
- [✓ 1, 5]
- ○ 1, 2, 6

Question 45:You have a need to transform a column named ‘timestamp’ to a date format. Assume that the column ‘timestamp’ is compatible with format date. You have written the code block down below, but it contains an error. Identify and fix it. df.select(to_date(col("timestamp"), "MM-dd-yyyy").show() *

- ○ We need to add a format and it should be the first parameter passed to this function. df.select(to_timestamp(‘yyyy-dd-MM’, col("date")))
- [✓ Format is not correct. You need to change it to: df.select(to_date(col("timestamp"), ‘yyyy-dd-MM’))]
- ○ to_date() is not a valid operation. Proper function is toDate() and also we need to change the format. df.select(toDate(col("timestamp"), "yyyy-MM-dd").show()
- ○ to_date() is not a valid operation. Proper function is toDate() df.select(toDate(col("timestamp"), "MM-dd-yyyy").show()

Question 46:Choose the equivalent code block to:df.filter(col("count") < 2)Where df is a valid dataframe which has a column named count *

- ○ df.getWhere("count < 2")
- [✓ df.where("count < 2")]
- ○ df.where(“count is smaller then 2").show(2)
- ○ df.select("count < 2")
- ○ df.where(count < 2)

Question 47:Which of the following is correct for the cluster manager ? *

- ○ Executes code assigned to it.
- ○ Reports the state of the computation back to the driver.
- ○ It is interchangeable with job.
- ○ All of the answers are correct.
- [✓ Keeps track of the resources available.]

Question 48:Which of the following describes the relationship between cluster managers and worker nodes? *

- ○ A worker node is a Java Virtual Machine (JVM) running on an cluster manager.
- ○ There are always more cluster manager nodes than worker nodes.
- ○ A cluster manager is a Java Virtual Machine (JVM) running on a worker node.
- [✓ Cluster manager creates worker nodes and allocates resource to it.]
- ○ There are always the same number of cluster managers and worker nodes.

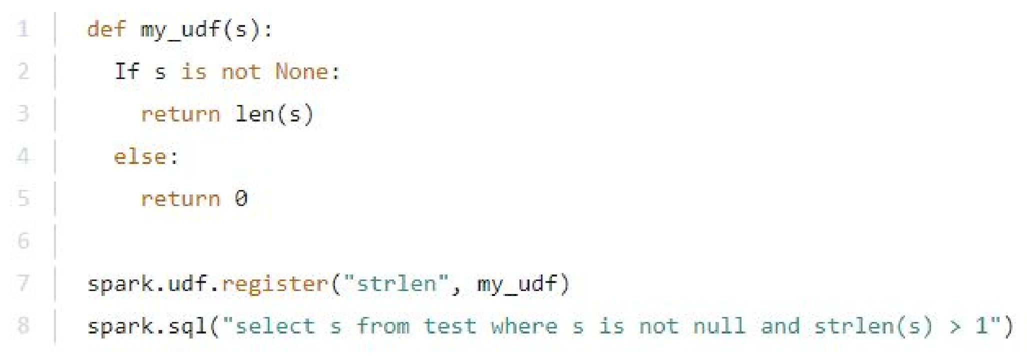
Question 49:We have an unmanaged table “my_table” If we run the code block down belowspark.sql(“DROP TABLE IF EXISTS my_table”) What will happen to data in my_table ? *

- ○ The data will be dropped also
- [✓ No data will be removed but you will no longer be able to refer to this data by the table name.]
- ○ This is not a valid code block
- ○ Spark will remove the table and also associated views

Question 50:Which of the following three operations are classified as a wide transformation ? Choose 3 answers: *

- [✓ filter()]
- [✓ orderBy()]
- drop()
- [✓ flatMap()]
- selectDistinct()
- coalesce(shuffle=true)

Question 51:Given the code block down below, a database test containing nulls, identify the error. *



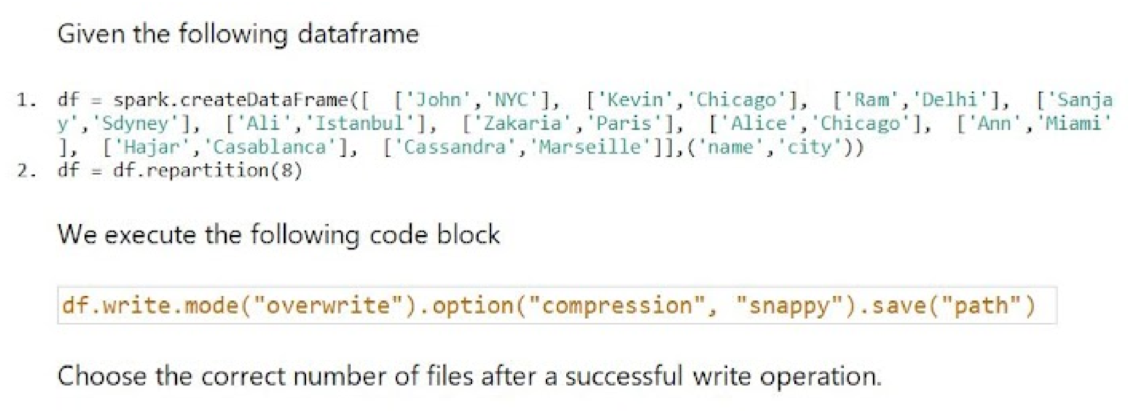
```
def my_udf(s):
  If s is not None:
    return len(s)
  else:
    return 0

spark.udf.register("strlen", my_udf)
spark.sql("select s from test where s is not null and strlen(s) > 1")
```

- [✓ This WHERE clause does not guarantee the strlen UDF to be invoked after filtering out nulls. So we will have null pointer]
- We need to use function ‘query’ instead of ‘sql’ to query table test.
- We need to create the function first and then pass it to udf.register
- There are no problems in this query.

Question 52:Your manager gave you a task to remove sensitive data. Choose the correct code block down below to remove name and city from the dataframe. df = spark.createDataFrame([['Josh','Virginia',25,'M'], ['Adam','Paris',34,'M']], ('name','city','age','gender')) *

- df.remove("name","city")
- df.drop("Josh","Adam")
- df.remove(name,city)
- df.drop(name,city)
- [✓ df.drop("name","city")]

Question 53: *



Given the following dataframe

```
df = spark.createDataFrame([  ['John','NYC'],  ['Kevin','Chicago'],  ['Ram','Delhi'],  ['Sanja
y','Sdyney'],  ['Ali','Istanbul'],  ['Zakaria','Paris'],  ['Alice','Chicago'],  ['Ann','Miami'
],  ['Hajar','Casablanca'],  ['Cassandra','Marseille']],('name','city'))
df = df.repartition(8)
```

We execute the following code block

```
df.write.mode("overwrite").option("compression", "snappy").save("path")
```

Choose the correct number of files after a successful write operation.

- 4
- [✓ 8]
- 1

Question 54:Which of the following operations can be used to create a new DataFrame with only the column “a” from the existing DataFrame df ? *

- ○ dataFrame.withColumnRenamed()
- ○ dataFrame.drop(“a”)
- ○ dataFrame.withColumn()
- [✓ dataFrame.select(“a”)]
- ○ ddataFrame.head()

Question 55:How make sure that dataframe df has 8 partitions given that it has 4 partitions ? *

- ○ df.setPartitition(8)
- ○ df.partition(8)
- ○ df.coalesce(8)
- [✓ df.repartition(8)]

Question 56:Which of the following code blocks returns a DataFrame with two new columns ‘a’ and ‘b’ from the existing column ‘aSquared’ where the values of ‘a’ and ‘b’ is half of the column ‘aSquared’ ? *

- ○ df.withColumn(aSquared, col(a) * col(a))
- ○ df.withColumn(“a”, col(“aSquared”)/2).withColumn(“b”, col(“aSquared”)/2)
- [✓ df.withColumn(“aSquared” /2 , col(a)).withColumn(“aSquared” /2 , col(b))]
- ○ df.withColumn(aSquared/2 , col(a)) .withColumn(aSquared /2 , col(b))
- ○ df.withColumn(“aSquared” /2 , col(“a”)).withColumn(“aSquared” /2 , col(“b”))

Question 57:What happens at a stage boundary in spark ? *

- ○ Application stops immediately .
- ○ Stage gets transformed to a job.
- [✓ At each stage boundary, data is written to disk by tasks in the parent stages and then fetched over the network by tasks in the child stage.]
- ○ Worker nodes restarts.

Question 58:Which of the following 3 DataFrame operations are classified as an action? Choose 3 answers: *

- [✓ limit()]
- foreach()
- first()
- show()
- printSchema()
- cache()

Question 59:When joining two dataframes, if there is a need to evaluate the keys in both of the DataFrames or tables and include all rows from the right DataFrame as well as any rows in the left DataFrame that have a match in the righ DataFrame also If there is no equivalent row in the left DataFrame, we want to instert null: which join type we should select ? df1.join(person, joinExpression, joinType) *

- joinType = “rightSemi”
- [✓ joinType = “rightAnti”]
- joinType = “right_outer”
- joinType = “rightOuter”

Question 60:Which of the following is true for an executor ? *

- ○ Worker nodes are synonymous with executors.
- ○ Executors nodes always have a one-to-one relationship with workers.
- [✓ Executors are the most granular level of execution in the Spark execution hierarchy.]
- ○ There could be multiple exectuors in a single worker node.